Aerospace Science and Technology 23 (2012) 93–107

Contents lists available at ScienceDirect

Aerospace Science and Technology

www.elsevier.com/locate/aescte

An interpolation tool for aerodynamic mesh deformation problems based on octree decomposition

M. Cordero-Gracia [a,*], M. Gómez [a], J. Ponsin [b], E. Valero [a]

[a] Universidad Politécnica de Madrid, School of Aeronautics, Pza. Cardenal Cisneros 3, E-28040 Madrid, Spain
[b] Instituto National de Técnica Aeroespacial, Fluid Dynamics Branch, Cta. Ajalvir, 4, E-28850 Madrid, Spain

ARTICLE INFO

Article history:
Received 1 June 2010
Received in revised form 5 April 2011
Accepted 15 June 2011
Available online 30 June 2011

Keywords:
Mesh motion
Octree decomposition
Radial basis functions

ABSTRACT

The necessity to modify a pre-existing computational mesh is a common requirement in many areas of computational fluid dynamics like aeroelasticity, optimization, etc. Here, we propose an approach to develop an efficient numerical mesh movement tool. The strategy relies on a three steps procedure: (i) generation of an octree decomposition of the geometry, (ii) definition of small interpolation domains, and (iii) application of local interpolation algorithms. Deformation is propagated from the moving boundaries towards the far field in a way similar to an advancing front methodology, which ensures continuity and numerical viability. The method can be applied to any type of mesh: structured, multiblock structured, unstructured and hybrid because it only uses geometric position of the mesh points, regardless of the particular mesh connectivities. The interpolation tool is based on radial basis functions. It will be showed that the method is very robust and generates a mesh with similar quality parameters as the original, it is computationally very efficient and can be easily parallelized.

1. Introduction

The ability to modify a pre-existing computational mesh is a common requirement in many areas of computational fluid dynamics (CFD) such as aeroelasticity and shape optimization. For instance, in the loosely-coupled aeroelastic methodology, the aerodynamic and structural models are computed on different meshes. CFD tools are applied to the aerodynamic mesh whereas computational structural mechanics methods (CSM) to the structural one. In order to connect both meshes, efficient numerical tools are necessary: 1) to transfer structural deformations to the aerodynamic mesh, and aerodynamic loads to the structural nodes, and 2) to deform the CFD flowfield mesh after a new aerodynamic surface has been computed. Another interesting field of application is aerodynamic shape optimization, where reliable and efficient algorithms for mesh movement are becoming of increasing importance. Whenever the optimizer makes a modification to the design variables controlling the geometric surface, the volume mesh must be adapted to reflect these changes.

In both problems, the mesh must be deformed/moved several times during the computation, which in the case of complex configurations, such as complete aircrafts, can increase the total computational time and degrade the quality of the solution. For these reasons, the mesh movement problem has gained increasing attention from the aeronautical industry.

The approaches for mesh deformation fall, broadly speaking, into two categories: pseudostructural and algebraic methods. The latter is commonly used in structured meshes. The most known of these methods is the transfinite interpolation (TFI) (Morton et al. [19], Liu et al. [15]), which can also be extended to multiblock structured meshes (Tsai et al. [25]). The former is commonly applied to unstructured meshes. This method can be understood as a variation of the elastic spring analogy (Batina [2,3], Bartels [1]), which considers the mesh as a network of linear springs and solves the static equilibrium equations to determine the new location of the mesh points. Its main disadvantage is that, in some situations, non-positive volume cells may appear and this requires that the different parameters involved in its definition must be carefully tuned. Different versions of the spring analogy method have been proposed by Farhat et al. [8], with the addition of torsional spring elements, Martineau and Georgala [17], with the addition of a rigid-body initialization (RBI) procedure; or Gao et al. [9], who use a nonlinear elastic boundary element method as deformation tools.

In any case, it is desirable that the mesh deformation tool fulfills the following requirements: i) Preservation of the connectivity of the original mesh, ii) ability to generate a valid deformed mesh for a wide range of perturbations of the boundaries, iii) applicability to a variety of mesh types: structured, multiblock structured, unstructured, hybrid, iv) mesh quality preservation, and v) efficiency in computational resources (time and memory).

* Corresponding author.
E-mail address: marta.cordero@upm.es (M. Cordero-Gracia).

1270-9638/$ – see front matter
doi:10.1016/j.ast.2011.06.002

Nomenclature

A_{as}	cross influence matrix
C_{ss}	interpolation matrix
C_{N_s}	cutoff of the interpolation centers number
G	coupling matrix
h	interpolation function
h_j	deformation of jth node
L	wing span
N_a	number of evaluation nodes
$N_{\max}$	maximum number of nodes that an octree can contain cube
$N_{s\max}$	maximum number of centers that an interpolation domain can contain
N_s	number of interpolation centers
$\bar{x}_j$	location vector of jth node
η	displacement at wing tip
γ	coefficients of the Π polynomial
ω	linear coefficients of the interpolation function
Φ	basis function
Π	polynomial of the interpolation function
φ	twist angle at wing tip

Superscripts

a	aerodynamic node
s	structural node

Subscripts

s	surface node
v	volume node

Despite the advances made on this topic during the last years, the problem of mesh movement is still open to new approaches and improvements. The CFD community demands more complex geometries every year and tries to solve more complex physics. Three-dimensional meshes of complete aircrafts with fine enough boundary layers, including flaps, nacelles, different gaps, etc., is commonplace nowadays. The numerical tools capable to cope with this kind of meshes are still under development.

The objective of this work is to apply a method based on interpolation with Radial Basis Functions (RBF) to mesh deformation problems regardless of how large the computational mesh is. Some previous attempts to apply this methodology to mesh data transfer have been realized by different authors: Cordero-Gracia et al. [6], Beckert and Wendland [4] and Wendland [27], studied with detail the application of different RBF versions to transfer deformations and loads between the structural and aerodynamic meshes. Rendall and Allen [20] have shown the feasibility of RBF when applied to mesh movement problems, however because of the global treatment of the mesh, its analysis is restricted to meshes with a small number of nodes. Recently, Rendall and Allen [21] have improved the efficiency of the RBF mesh motion method reducing the number of surface points used as centers of the interpolation. On the other hand, Jakobsson and Amoignon [11] making use of a similar methodology tested a wide range of basis functions in order to applicate the interpolation scheme in optimization process applied to a fixed-wing problem. Michler [18] has developed an adaptive process to select what surface nodes are the best to act as centers for the interpolants, and apply RBF based mesh deformation to an aircraft control surface deflection (aileron and horizontal tail).

Here, we will focus on the extension of these methods to apply RBF to the deformation of general CFD volumetric mesh, regardless of the number of nodes and connectivities. The strategy consists in the definition of a reasonable number of interpolation domains that cover the whole mesh. Each domain represents an isolated deformation transfer problem that can be solved independently from the other domains using a suitable interpolation function. Those interpolation domains are efficiently generated and managed with the aid of an octree structure and the interpolation is carried out from the boundary walls towards the far field in a way that mimics an advancing front methodology. This method has two main advantages: it reduces the memory requirements associated with global interpolation methods, such as volume splines, being computationally more efficient for meshes of a large number of nodes and, finally, it can be applied directly to any type of mesh (hybrid, unstructured, structured multiblock) because it does not require any connectivity information. A numerical mesh deformation tool, called *MeshMove* has been developed as part of this work.

The paper is organized as follows. In Section 2 a brief mathematical description of the interpolation method is outlined. The application of these methods to general volume mesh movement problems is developed in Section 3. Some numerical results are discussed in Section 4.

2. Interpolation techniques

Although many different techniques exist for interpolating a finite set of values, the selection of one is strongly influenced by the requirements imposed by the underlying physical or mathematical problem. Common desirable properties of interpolation techniques are robustness, efficiency and monotonicity, the last property being of special importance in order to avoid spurious oscillations in the continuous function to be reconstructed, and hence in the interpolated values.

Within an aeronautical framework, several interpolation methods have been developed to exchange information between points of different meshes. Smith et al. [23] have reviewed six different schemes for data transfer, most of them are based on the splines methodology. The methods were evaluated with accuracy and robustness criteria. According to their conclusions, surface splines are the most accurate and robust, but must be implemented as local methods in real 3D tests. Hounjet and Meijer [10] expose the mathematical description of different interpolation methods, making the distinction between planar and non-planar configurations. Among them, the so-called volume spline function (VSF, a particular case of RBF), stands out as a very simple interpolator that can be easily applied to any 3D data.

This interpolation scheme satisfies the positivity constraint which prohibits the occurrence of local extrema and is suitable to be applied to a set of mesh points without the necessity to know the connectivity among them (in case of unstructured meshes) or the topology information (in case of multiblock structured meshes). They can be viewed as a mathematical operation in which the deformation of a set of nodes, called centers, produces the displacement of another set of nodes, called evaluation nodes. The deformation of a volumetric mesh requires the application of a local interpolation technique. The locality properties of the interpolation technique can be achieved via either, by interpolation over the entire mesh with a global interpolant function with compact support or application of a global interpolant function on small local domains of the initial mesh.

2.1. Mathematical formulation

Given a finite set of d-dimensional centers $X^s = \{\bar{x}_1^s, \ldots, \bar{x}_{N_s}^s\}$ and its displacement field $H^s = \{h_1^s, \ldots, h_{N_s}^s\}$, the aim is to obtain the corresponding smooth and regular displacement field $H^a = \{h_1^a, \ldots, h_{N_a}^a\}$ for the set of evaluation nodes $X^a = \{\bar{x}_1^a, \ldots, \bar{x}_{N_a}^a\}$. To preserve the notation used in CFD–CSM coupling problems, we still denote with the superscripts ^{s} for the centers or structural nodes, and ^{a} for evaluation or aerodynamic nodes.

The interpolation method consists in building a continuous spatial function $h(\bar{x})$, using the discrete values $\bar{x}_i^s$, defined as:

$$h(\bar{x}) = \sum_{i=1}^{N_s} w_i \Phi\left(\left\|\bar{x} - \bar{x}_i^s\right\|\right) + \Pi(\bar{x}), \tag{1}$$

where Φ is a fixed kernel function which is radial with respect to the Euclidean distance,

$$\|\bar{x}\| = \sqrt{x_1^2 + \cdots + x_d^2}, \tag{2}$$

and Π is an optional low degree d-variate polynomial. The weights w_i are calculated by requiring exact recovery of the original function over the centers,

$$h_i^s \equiv h\left(\bar{x}_i^s\right) \quad \forall i = 1, \ldots, N_s. \tag{3}$$

If the polynomial Π is included, the system must be completed with the additional zero condition:

$$\sum_{i=1}^{N_s} w_i q(\bar{x}_i) = 0, \tag{4}$$

for all polynomials q with degree $\deg(q) \leqslant \deg(\Pi)$.

The interpolation scheme developed in the present work only considers linear polynomials Π. This condition is sufficient to guarantee the uniqueness of the solution, provided that a conditionally positive definite function Φ of order $m \leqslant 2$ is used (for additional details, Beckert and Wendland [4]). In this way, and focusing on the 3D case, we define:

$$\Pi(\bar{x}) = \gamma_0 + \gamma_1 x + \gamma_2 y + \gamma_3 z, \qquad \bar{x} = (x, y, z). \tag{5}$$

Now, the zero condition (4) is expressed as:

$$\sum_{i=1}^{N_s} w_i = 0, \qquad \sum_{i=1}^{N_s} w_i x_i^s = 0, \qquad \sum_{i=1}^{N_s} w_i y_i^s = 0,$$
$$\sum_{i=1}^{N_s} w_i z_i^s = 0. \tag{6}$$

Assembling (3) and (6), we obtain the following system of equations for the weights w_i and the polynomial coefficients γ_k

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ h_1^s \\ h_2^s \\ \vdots \\ h_{N_s}^s \end{pmatrix} = \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 1 & 1 & \ldots & 1 \\ 0 & 0 & 0 & 0 & x_1^s & x_2^s & \ldots & x_{N_s}^s \\ 0 & 0 & 0 & 0 & y_1^s & y_2^s & \ldots & y_{N_s}^s \\ 0 & 0 & 0 & 0 & z_1^s & z_2^s & \ldots & z_{N_s}^s \\ \hline 1 & x_1^s & y_1^s & z_1^s & \Phi_{s_1 s_1} & \Phi_{s_1 s_2} & \cdots & \Phi_{s_1 s_{N_s}} \\ 1 & x_2^s & y_2^s & z_2^s & \Phi_{s_2 s_1} & \Phi_{s_2 s_2} & \cdots & \Phi_{s_2 s_{N_s}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N_s}^s & y_{N_s}^s & z_{N_s}^s & \Phi_{s_{N_s} s_1} & \Phi_{s_{N_s} s_2} & \cdots & \Phi_{s_{N_s} s_{N_s}} \end{array}\right) \times \begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ w_1 \\ w_2 \\ \vdots \\ w_{N_s} \end{pmatrix}, \tag{7}$$

or equivalently, in matrix notation form,

$$\boldsymbol{U}_s = C_{ss}\boldsymbol{\omega} \tag{8}$$

with the symmetric matrix C_{ss} being the interpolation matrix, and $\Phi_{s_j s_i} = \Phi(\|\bar{x}_j^s - \bar{x}_i^s\|)$. The formulation (8) is valid only for a scalar displacement field $\{h_i^s\}$. Its extension to higher dimensions (vectorial displacement) is straightforward by employing (8) to each coordinate direction, and taking into account that C_{ss} does not vary with the directions.

Now, we can compute the displacement of the aerodynamic nodes by applying (1) to the evaluation nodes X^a, which gives:

$$\begin{pmatrix} h_1^a \\ h_2^a \\ \vdots \\ h_{N_a}^a \end{pmatrix} = \begin{pmatrix} 1 & x_1^a & y_1^a & z_1^a & \Phi_{a_1 s_1} & \Phi_{a_1 s_2} & \cdots & \Phi_{a_1 s_{N_s}} \\ 1 & x_2^a & y_2^a & z_2^a & \Phi_{a_2 s_1} & \Phi_{a_2 s_2} & \cdots & \Phi_{a_2 s_{N_s}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N_a}^a & y_{N_a}^a & z_{N_a}^a & \Phi_{a_{N_a} s_1} & \Phi_{a_{N_a} s_2} & \cdots & \Phi_{a_{N_a} s_{N_s}} \end{pmatrix} \times \begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ w_1 \\ w_2 \\ \vdots \\ w_{N_s} \end{pmatrix}, \tag{9}$$

with $\Phi_{a_j s_i} = \Phi(\|\bar{x}_j^a - \bar{x}_i^s\|)$. Expressed in compact matrix form notation yields:

$$\boldsymbol{U}_a = A_{as}\boldsymbol{\omega}. \tag{10}$$

From a computational point of view, solving the coupled systems (8) and (10) can be made following two different approaches: (a) calculation of a coupled matrix, or (b) resolution of an equivalent linear problem.

1. Coupling matrix computation
   From the linear system (8), it is possible to obtain the vector of unknowns $\boldsymbol{\omega} = C_{ss}^{-1}\boldsymbol{U}_s$ which, when it is substituted in (10), yields a direct relation between the displacements of the evaluation nodes and centers, namely

   $$\boldsymbol{U}_a = A_{as}C_{ss}^{-1}\boldsymbol{U}_s = G\boldsymbol{U}_s. \tag{11}$$

   Here, $G = A_{as}C_{ss}^{-1}$ is defined as the coupling matrix. The matrix G has the advantage that it is the same for all displacement directions, however its evaluation is very costly in terms of memory and computational time. If the number of structural points (N_s) is bigger than a certain (computer dependent) threshold, the interpolation matrix inversion could be unfeasible even for simple problems.
2. Solving a linear system
   From a numerical point of view, it is more efficient to compute the coefficient vector $\boldsymbol{\omega}$ by solving the system (8), instead of computing the inverse of the interpolation matrix C_{ss}. If the size of the interpolation matrix is not too large, the solution

Table 1
Radial basis functions.

Function	Definition ($\Phi(\bar{x})$)
Volume spline	$\|\bar{x}\|$
Wendland C^0	$(1-\|\bar{x}\|)_+^2$
Wendland C^2	$(1-\|\bar{x}\|)_+^4(4\|\bar{x}\|+1)$
Wendland C^4	$(1-\|\bar{x}\|)_+^6(35\|\bar{x}\|^2+18\|\bar{x}\|+3)$

of the system (8) can be efficiently performed by any numerical linear algebra package like *LAPACK* [28]. As we will see later, it is necessary to limit the size of the interpolation matrix in order to keep the required interpolation time within an acceptable range. The size of the interpolation matrix will depend on the method selected to solve the system. Once the vector $\boldsymbol{\omega}$ has been calculated, the new displacement field $\boldsymbol{U}_a$ is quickly and easily computed by the matrix–vector multiplication given by (10). However, from a practical point of view, it is recommended not to allocate the memory for the matrix A_{as}, instead, it is possible to perform a line by line product, thus reducing the memory requirements.

2.1.1. Note on the definition of the polynomial (5) to the mesh movement problem

The systems (8) and (10) constitute the original volume spline formulation defined to perform CFD–CSM interpolations (displacements or loads). In this case, the inclusion of a polynomial term (5), at least of first order, is mandatory to ensure conservation of total force and moment. However, in order to transfer deformations from aerodynamic nodes to the rest of the volume mesh nodes (the mesh movement problem), those linear terms have to be excluded. Otherwise, the linear polynomial $\Pi(\bar{x})$ will produce the transfer of fictitious displacements to the far field boundary nodes which have no physical sense (Rendall and Allen [20]). Therefore, for the volume mesh movement problem, the polynomial reduces to

$$\Pi(\bar{x}) = \gamma_0, \tag{12}$$

and the zero condition is simplified to:

$$\sum_{i=1}^{N_s} w_i = 0. \tag{13}$$

2.2. Radial basis functions

To close the definition of the interpolant (1), the kernel ($\Phi(\|\bar{x}\|)$) is selected from the family of Radial Basis Functions (Buhmann [5]). RBFs are valid for any dimensional interpolation problem, are both translation and rotation invariant, many of them generate a positive definite interpolation matrix and only depend on the distance between the nodes. In order to investigate their influence in the resulting mesh deformation tool, some of the most common RBFs used in the CFD mesh movement context has been compared. As it will be shown in Section 4, the strategy proposed for performing the interpolation makes the results quite independent of the kind of RBF selected for the interpolation. In Table 1 and Fig. 1, we show the most representative RBFs incorporated in the mesh movement tool, which range from the basic volume spline function, with a global character, to the smooth compactly supported radial functions proposed by Wendland [26].

3. Mesh movement methodology

The interpolation procedure previously described is the kernel of the mesh movement methodology. In fact, it is possible to apply this method as a global interpolant tool to a mesh movement problem. In that case, the matrix C_{ss} includes all the mesh nodes whose deformation is going to be transfered, a fact that limits the feasibility of the method only to meshes with a number of centers limited by the processor capacity. As the matrix A_{as} is not stored, and the linear system to be solved scales with $(N_s \times N_s)/2$, the size of the meshes, that can be deformed with this method, will depend on the computer resources and the linear system solver.

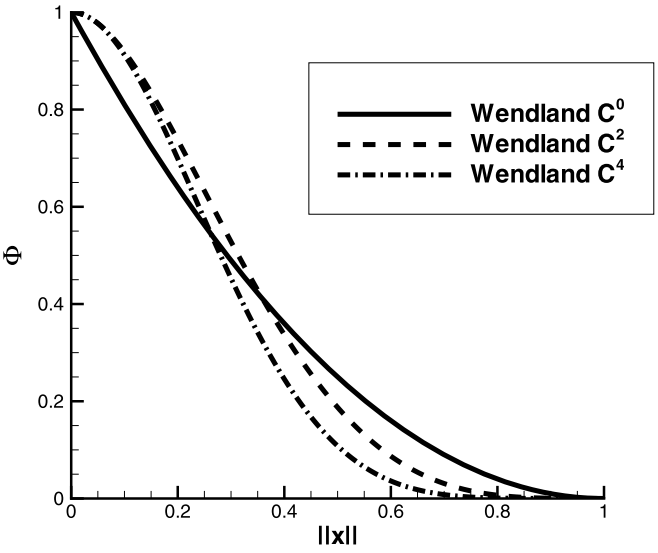

Fig. 1. Wendland radial basis functions.

Applying this global interpolation method, such as the volume spline function, to meshes with a large number of nodes requires huge computational resources in term of memory and CPU time. The use of RBF with local compact support can alleviate this problem because only the nodes located in the compact support of the function will appear in the interpolation (see Fig. 1). However, it is necessary to define new strategies to cope with real 3D configurations. The idea followed in this work was originally introduced by Wendland [27], who proposed to combine the radial function approach with a localization technique called “partition of unity” and devised the necessity to include an “intelligent” data structure for generating smaller domains in which the interpolation could be applied locally. He applied a preliminary version of this method to perform a CFD–CSM surface interpolation obtaining important savings in terms of memory and CPU time compared to global interpolation methods.

Therefore, the key aspect of the methodology is the interpolation strategy. The strategy should be designed so as to take into account two basic aspects: the number of the nodes involved and the degree of proximity between the centers and the evaluation nodes. The latter is related to the computational efficiency, while the former to the accuracy of the solution.

Following these guidelines, we have developed an efficient numerical mesh movement tool based on a local multi-domain strategy. Each domain represents an isolated deformation transfer problem that can be solved independently from the others while maintaining the accuracy of the interpolation. This can be accomplished by selecting a minimum number of center points per domain, but not too large to avoid excessive computational overhead originated by the matrix operations associated with the interpolation method.

The domains are generated with the aid of an octree-data structure. The interpolation is applied to each domain in a sequence which starts on the wall boundary nodes and spreads out as a front, reaching the far-field boundary nodes at the end of the interpolation sequence. The proposed methodology is independent of the mesh type due to the fact that no connectivity information is required. Only the node coordinates are required to perform the

```
Input:  aerodynamic mesh geometry (N_s nodes),
        volume mesh geometry (N_v nodes),
        maximum number of nodes within a cube (N_max)
Output: Octree
Generates the first cube
for each node from 1 to N_s + N_v do
    do
        search for the cube which the node belongs to
        if that cube contains less than N_max nodes
            the node is introduced in that cube
            exit
        else
            splits the cube in 8 children cubes
            distributes the N_max nodes among the children
        end if
    end do
end for
```

Fig. 2. Octree generation algorithm.

whole interpolation process. This constitutes an advantage when comparing to other methods used for CFD–CSM coupling, such as elasticity methods, whose require information about the mesh connectivity. The mesh movement is generated by using as input the deformation of a cloud of nodes (centers), resulting as output the deformation of another cloud of nodes (evaluation nodes).

3.1. Detailed description of the interpolation strategy

The devised strategy is marked by the local character of the interpolator. In a first step, interpolation domains are defined as non-necessarily disjoint sets comprising centers and evaluation nodes linked by their degree of proximity, and such that their union covers the whole mesh. The second step amounts to establishing an optimal ordering to the domain deformation sequence. The underlying physics of the problem imposes as a natural condition to start from those domains which contain centers (nodes situated on the body surface). Once the first layer surrounding the body has been deformed, the mesh deformation proceeds to the next layers in a sequential fashion in which previously deformed evaluation nodes act as interpolation centers for the next layer. With this sequential procedure the shape of the body is being reproduced in each layer.

Each interpolation domain defines a complete interpolation problem. A domain is characterized by the following data: (a) the identifiers of both the centers and the evaluation nodes which build up the C_{ss} and A_{as} matrices of the corresponding interpolation, and (b) an index which represents its position within the ordered domain deformation list.

3.1.1. Generation of the interpolation domains

Quadtrees and octrees have been used extensively for 2D and 3D mesh generators (Löhner [16], Knuth [14]). Octree data structures have become an efficient search algorithm for arbitrary point distributions. The octree generation algorithm starts by defining the smallest cube containing the whole mesh. This cube is recursively subdivided into eight equally-sized children cubes, each of which contains at most $N_{\max}$ points. At the end of the process a tree structure of cubes has been generated, some of which may be empty. The pseudocode for the octree generation algorithm is shown in Fig. 2.

Using the Quadtree generation algorithm explained by Löhner [16] as a starting point, we have implemented a dynamic octree structure stored in a linked list where quick access to the information is crucial. Each element of the linked list contains all the relevant information of a cube: geometric information (coordinates of two opposite corners), hierarchic information (level of depth within the octree, parent cube identifier, position within the parent cube, etc.) and content information (labels of the nodes it contains or, that lacking, the identifiers of their children).

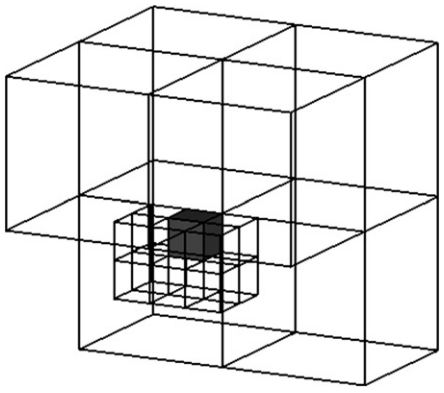

Fig. 3. Sketch of a kernel cube whose nodes are going to be deformed, and its surrounding neighbour cubes.

Depending on their role in the interpolation process, the mesh nodes can be classified on the following way:

(a) Surface nodes. Nodes of the aerodynamic mesh whose deformation has to be transferred to the volume mesh. They always act as interpolation centers, even those with zero deformation.
(b) Far field nodes. Nodes of the volume mesh which correspond to the far field. The deformation of these nodes is zero. They act as interpolation centers to preserve the continuity of the deformation field.
(c) Contour nodes. Nodes of the volume mesh that coincide with any surface node. For the sake of coherence, these nodes are assigned the same deformation as their counterparts on the aerodynamic mesh. Thus, they don't take part in the interpolation process.
(d) Volume nodes. Nodes of the volume mesh whose deformation has to be computed. In a global interpolation strategy, they always act as evaluation nodes. However, in the local strategy, once they have been deformed, they can act as centers in subsequent interpolation domains.

In summary each childless octree cube, a so-called leaf or external cube, constitutes a computational unit. An interpolation domain consists of a computational unit containing the volume nodes that are going to be deformed (defined from now on as kernel

```
Input:  Octree
Output: Interpolation domains (label of each kernel and its neighbours).
        Interpolation sequence array of indices.
Extracts the external cubes (those with no children)
Calculates the (i, j, k) coordinates of the external cube
for each external cube do
    Obtains its neighbour external cubes list
    Counts domain's surface nodes
end for
# Obtains the head domain:
Looks for the domain containig the maximum surface nodes,
  and which kernel has volume nodes
# Calculates the first layer, starting with the head domain:
Identifies what kernel has surface nodes inside
for each of these kernels do
    if it is a neighbour of a kernel that has already been included in the list
        if it isn't a member of that list
            it is added to the list
        end if
    end if
end for
# Calculates the advancing layers:
do
    for each kernel of the previous layer do
        arrange its neighbours: according to their size and
        to their number of volume nodes
        for each arranged neighbour do
            if it isn't a member of the list
                it is added to the list
            end if
        end for
    end for
end do
```

Fig. 4. Algorithm to construct the sequence of interpolation domains.

cube) and all its neighbour external cubes that contain any kind of centers. A neighbour cube of the kernel is an octree cube which shares a face, edge or vertex with it (Fig. 3). There are as many interpolation domains as kernel cubes in the octree. Once performed the interpolation of the nodes inside a kernel cube, they can act as centers for other neighbour kernels not yet interpolated. This strategy guarantee the transfer of the deformation in a smooth manner across the mesh.

For the identification of the neighbour cubes of a given kernel, a search algorithm has been developed that scans the cubes of the linked list using only integer arithmetics, without the need to resort to geometric computations. The idea is to take advantage of the topological structure of the octree to generate a scale in each of the three spatial directions. The unit on each coordinate axis being the edge length of the smallest cube in the octree. In this way, any cube in the octree can be localized by means of the integer coordinates of two opposite vertices (i_0, j_0, k_0) and (i_0+n, j_0+n, k_0+n), n being an integer that depends on the cube size. The identification of the neighbours comes down simply to a search within the range $i_0 \leqslant i \leqslant i_0+n$, $j_0 \leqslant j \leqslant j_0+n$ and $k_0 \leqslant k \leqslant k_0+n$.

3.1.2. Domain interpolation sequence

Once the domains have been generated, it is necessary to define an appropriate sequence of interpolation transfer in order to make the process robust and accurate. Due to the local interpolation character, the deformation in each domain will depend on which nodes are selected as center nodes. The selection of those nodes will depend also on the order in which each domain interpolation is performed.

The first layer in the deformation sequence is composed by those domains which contain boundary surface nodes. The process will start in the domain that contains the maximum number of centers for performing the interpolation. The next domains to be interpolated are the neighbours of the initial domain and the process is continuing until all the domains belonging to the first layer are interpolated. The second layer is composed by those domains which are neighbours of the domains of the first layer and have not been interpolated yet. This step can be viewed as if the first layer acts as a boundary for the interpolation of the second layer, because the volume nodes of the first layer are centers for the interpolation of the second layer. In this way, the continuity of the deformed field is guaranteed as we move through the advancing front sequence of domains. Fig. 4 details the algorithm used to define the interpolation sequence of the domains.

Once the sequence of the interpolation has been defined, it is necessary to define, for each domain, which are the centers and evaluation nodes. Fig. 5 outlines the algorithm used to perform this task. In terms of memory, the computational requirements are kept low enough by means of the user-defined parameter $N_{s\max}$. This parameter limits the dimension of the interpolation matrix C_{ss} to affordable values.

In practice, the interpolation process is often performed in several sweeps. Sometimes this is mandatory, e.g. in full unsteady simulations where the boundary is being deformed in each time step. In other situations, the sweeps are necessary to make the process robust. Such situations occur when large surface deformations arise, for example in the ice accretion problem. In this case, the interpolation process must be decomposed into several

```
Input:  Octree,
        Ordered list of the interpolation domains.
Output: For each domain, label's array of centers and evaluation nodes
for each interpolation domain i from 1 to N_dom do
    # Output nodes
    Defines labels array of kernel's volume nodes (evaluation nodes)
    # Input nodes:
    Defines labels array of domain's surface nodes
    if input nodes ≤ N_smax
        add domain's farfield nodes to the input array
    end if
    if input nodes ≤ N_smax
        for each deformed neighbour cube do
            add its volume nodes to de input array
            if (input nodes > N_smax) exit
        end for
    end if
end for
```

Fig. 5. Algorithm for defining the centers and evaluation nodes.

deformation steps. At each time step a new interpolation must be calculated according to the following scheme.

1. An octree is generated from both volume and boundary mesh nodes.
2. An ordered list of the cubes containing volume nodes is performed according to the criteria already given.
3. Given a cube of the former list, its corresponding input–output data information is stored.
4. The interpolation is carried out.
5. The mesh nodes positions are updated.
6. A mesh quality test is carried out after each interpolation.

From a computational point of view, two different aspects must be considered:

1. In order to adopt a compromise between the number of interpolation domains and the maximum size of the interpolation matrix in each domain, two different parameters have been defined:
   - $N_{\max}$ is the maximum number of nodes within an octree cube. Its value controls the number of total external cubes in the octree, which is related with the preprocessing time (steps 1 to 3).
   - $N_{s\max}$ is the maximum number of center nodes within an interpolation domain. Its value controls the size of the interpolation matrix C_{ss}, which is related with the evaluation time (step 4).
2. It is recommendable to apply several tests to measure the quality of the mesh resulting from the deformation. In this work, we have used two quality metrics (described in Section 4.2.1) not only to prove the validity of the methodology, but also as a criterion for stopping the computation. The maximum allowed deformation that can be transferred to the volume mesh is reached when any of the quality metric parameters go below a prescribed threshold which signals the presence of degenerated cells in the mesh.

4. Numerical results

In this section three different numerical tests are proposed to check the validity of the method. Firstly, the comparison between the CPU time of global and local interpolation methods has been realized in a simple structured mesh. The effect of the number of surface nodes is studied. Secondly, two more realistic configurations are treated. The first one is an unstructured and non-viscous mesh composed of tetrahedrons with around 34 000 surface nodes. In this case global interpolation is not possible (in our current computer capabilities), but local interpolation can be performed without problems even for very large deformations. The last test involves a typical viscous mesh of a wing configuration, where it is showed that the quality of the deformed mesh is maintained within acceptable values.

4.1. Validation test: computational cost

In this case, we compare both local and global methodologies, by applying the same tool, *MeshMove*, to the same configuration. This comparison has been carried out by generating, based on a cylinder, a set of meshes with increasing number of surface nodes. The original surface mesh is a cylinder of unit radius ($x^2 + y^2 = 1$, $0 \leqslant z \leqslant 10$) which is deformed (Fig. 6(b)) in only one step to become an elliptical cylinder ($x^2 + 4y^2 = 1$, $0 \leqslant z \leqslant 10$) in such a way that for any center (x^s, y^s, z^s), the displacement is given by $\boldsymbol{h}^s = (0, -0.5y^s, 0)$. Both approaches, local and global, were executed by an appropriate selection of the parameters $N_{\max}$ and $N_{s\max}$. In the local analysis, the control parameters are taken as $N_{\max} = 100$ and $N_{s\max} = 1200$, whereas for the global one $N_{\max} = N_s + N_v$ and $N_{s\max} = N_s$. The computations were performed on an Intel Xeon E5620 processor with 12 GB of RAM memory and 2.4 GHz of clock speed, the code *MeshMove* was written in Fortran 95 and compiled with the GNU Fortran compiler. The GNU linear algebra package – LAPACK – was used for solving the symmetric linear system (8).

The CPU time required for both methodologies is shown in Fig. 7(a). It is observed that there exists a threshold number of interpolation centers, C_{N_s}, above which the local strategy is more efficient in terms of CPU time. Moreover, the local strategy is far less limited in terms of the memory required to perform the interpolation. This result is the expected one taking into account that the solving time of a direct method, as the one used by the LAPACK library, scales with the number of centers to the third power. In addition, Fig. 7(b) shows a close up of both curves near their intersection point $C_{N_s} \approx 6000$. In Table 3 is detailed the preprocessing and evaluation time for each of the test cases here studied. It is observed that the preprocessing time takes around 3% of the total computational time. This fact shows that the local strategy interpolation time principally depends on the number of interpolation domains, therefore different meshes but with similar number of domains will give similar computational times. In the present test, the surface mesh ranges from 3000 to 30 000 nodes, but the

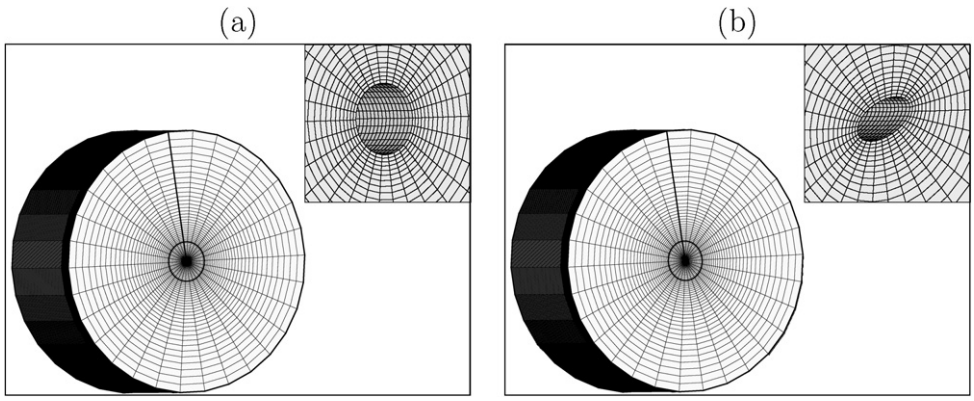

Fig. 6. Validation test: (a) Original structured cylindrical mesh; (b) deformed mesh.

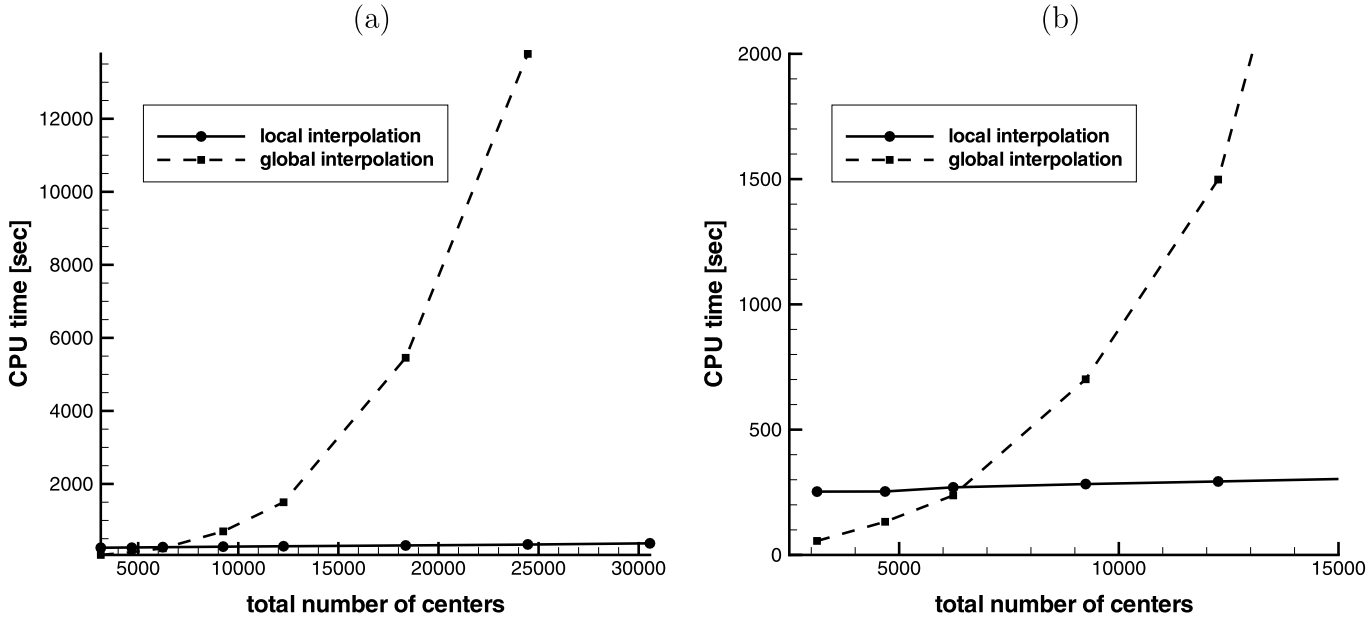

Fig. 7. (a) CPU time comparison between domain strategy (local interpolation) and direct application (global interpolation); (b) close up near the intersection point of the two curves.

number of domains is nearly constant (about 12 000), giving not significant differences of the interpolation time. As we will see later, the number of interpolation domains is strongly affected by the kind of mesh. For viscous meshes, where the nodes are clustering close to the surface, the number of domains is increased, hence the total deformation time.

4.2. Validation test: smoothness and robustness

4.2.1. Mesh quality metrics

In this section a quality assessment of the meshes obtained with the new approach is presented. To evaluate the quality of the resulting meshes, algebraic metric parameters based on the Jacobian and related parameters (Knupp [12]) are used. These parameters provide a measure of the quality of the mesh elements in terms of their size (volume), orientation, shape and skewness.

The main goal of a mesh movement method is to preserve the quality of the initial mesh or, at least, to minimize the quality degradation throughout the deformation process. It is supposed that the initial mesh has an acceptable quality and the resulting mesh should have a similar quality after deformation.

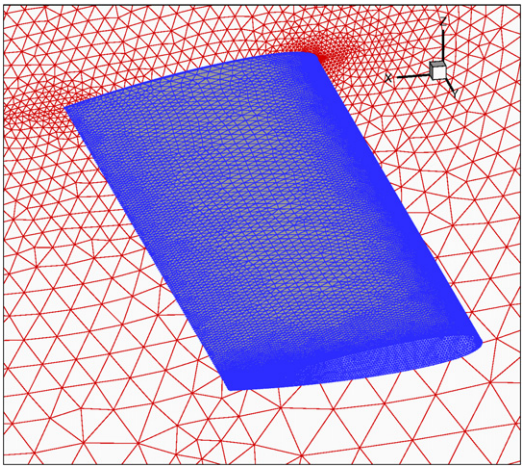

Fig. 8. NACA0012 wing.

The ratio between the current and initial element volume, τ, can be used to measure the change in element size. The *relative size metric* (Knupp [12]), defined as $f_{size} = \min\{\tau, 1/\tau\}$, reaches its

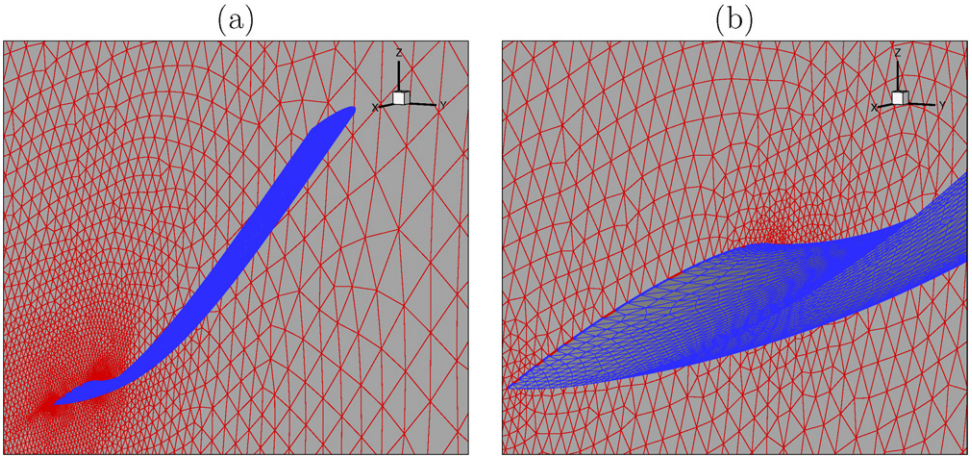

Fig. 9. NACA0012 wing: (a) result of the bending test (100%L displacement at tip); (b) close up of the wing near the root.

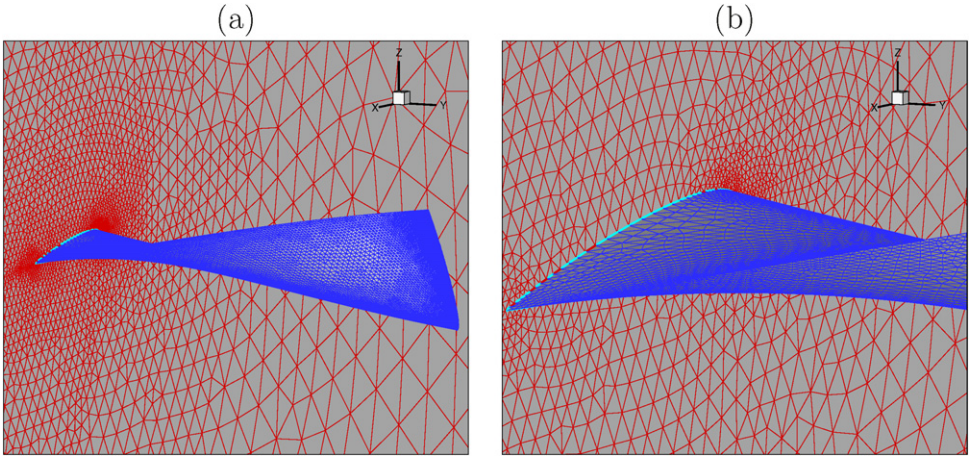

Fig. 10. NACA0012 wing: (a) result of the torsion test (70° twist at tip); (b) close up of the wing near the root.

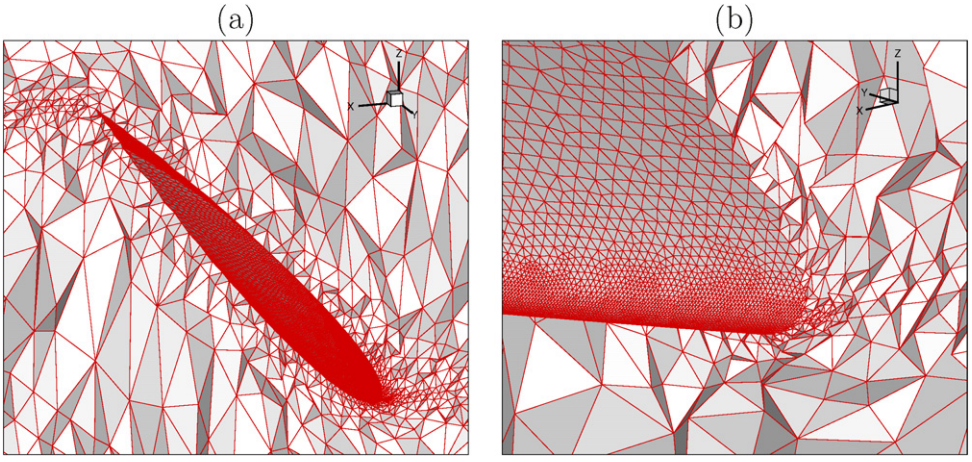

Fig. 11. NACA0012 wing: (a) deformed mesh due to 70° twist; (b) close up at 80% of the span and the wing leading edge.

maximum value, $f_{size} = 1$, when the volume of an element has not changed after the deformation. A negative value of this parameter can be obtained if the deformed element is degenerated. Changes in the shape of the element can be measured with the *shape metric*, f_{shape} (Knupp [13]), based on the Frobenius matrix norm. This parameter is a combination of the *skew metric*, which measures the element distortion (de Boer et al. [7]), and element edge-length ratios (Knupp [13]). The range of this parameter is $0 \leqslant f_{shape} \leqslant 1$, being 1 if the element shape is a scaled rotation of the ideal element shape and 0 if, and only if, the three edges at one vertex are coplanar.

In this work, the average and minimum values of f_{size} and f_{shape} over the entire mesh have been taken as representative values of the mesh quality. The higher the average value of the metric, the more stable, robust and accurate the result will be. The minimum value of the metric must be larger than zero in order to avoid degenerated cells whose influence on the stability and accuracy of the computations is very negative.

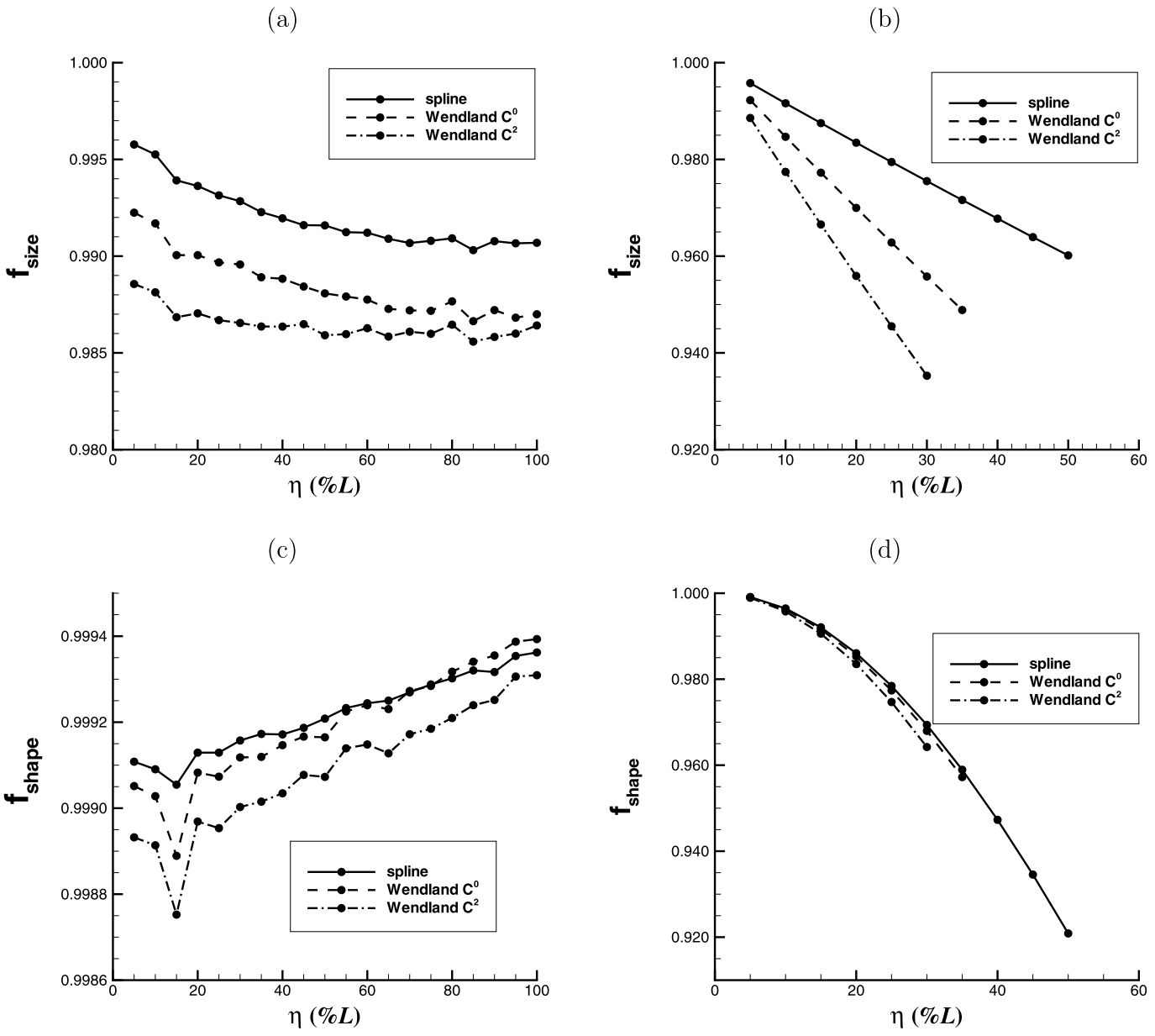

Fig. 12. NACA0012 validation test. Mean values of quality parameters for bending test: (a) and (c) multi-step deformation; (b) and (d) one step deformation.

4.2.2. Inviscid mesh

An inviscid tetrahedral mesh of 180 101 nodes is considered here to assess the mesh deformation tool. The mesh configuration corresponds to a NACA0012 rectangular wing with aspect ratio of 6.0. The surface mesh has 34 007 nodes (Fig. 8). Owing to the relatively high number of surface nodes, only the local strategy is feasible in this test case.

Two types of wing deformation have been considered: torsion and bending. They would result from the application of a constant torque and a uniformly distributed load, respectively, assuming that the wing behaves structurally like a simple beam located at quarter-chord. Considering a wing of span L and with the y-axis oriented spanwise from the root towards the tip, the mean line vertical displacement for the bending, $\eta(y)$, can be expressed as

$$\eta(y) = \frac{y^2(6L^2 - 4Ly + y^2)}{3L^4}$$

and the twist angle $\varphi(y)$ induced by the torsion is computed as

Table 2
Maximum deformation reached when running *MeshMove* tool over NACA0012 wing in one step and iteratively.

Test	Spline	Wendland C^0	Wendland C^2
$\varphi_{\max}$ in 1 step	51°	37°	34°
$\varphi_{\max}$ in multiple steps	100°	65°	50°
$\eta_{\max}$ in 1 step	54%L	38%L	34%L
$\eta_{\max}$ in multiple steps	100%L	100%L	100%L

$$\varphi(y) = \frac{y}{L}\varphi_{\max}.$$

For each of the two types of deformation, two kinds of robustness tests were run: first, the maximum deformation reached by running the algorithm once with the stopping criteria that either f_{size} or f_{shape} are outside their limits. Second, the deformation achieved by running the algorithm iteratively. In the latter case, at each step the twist at wing tip, φ, is increased by 5° for torsion, while the vertical displacement at wing tip, η, by 5% of the span, L, for bending.

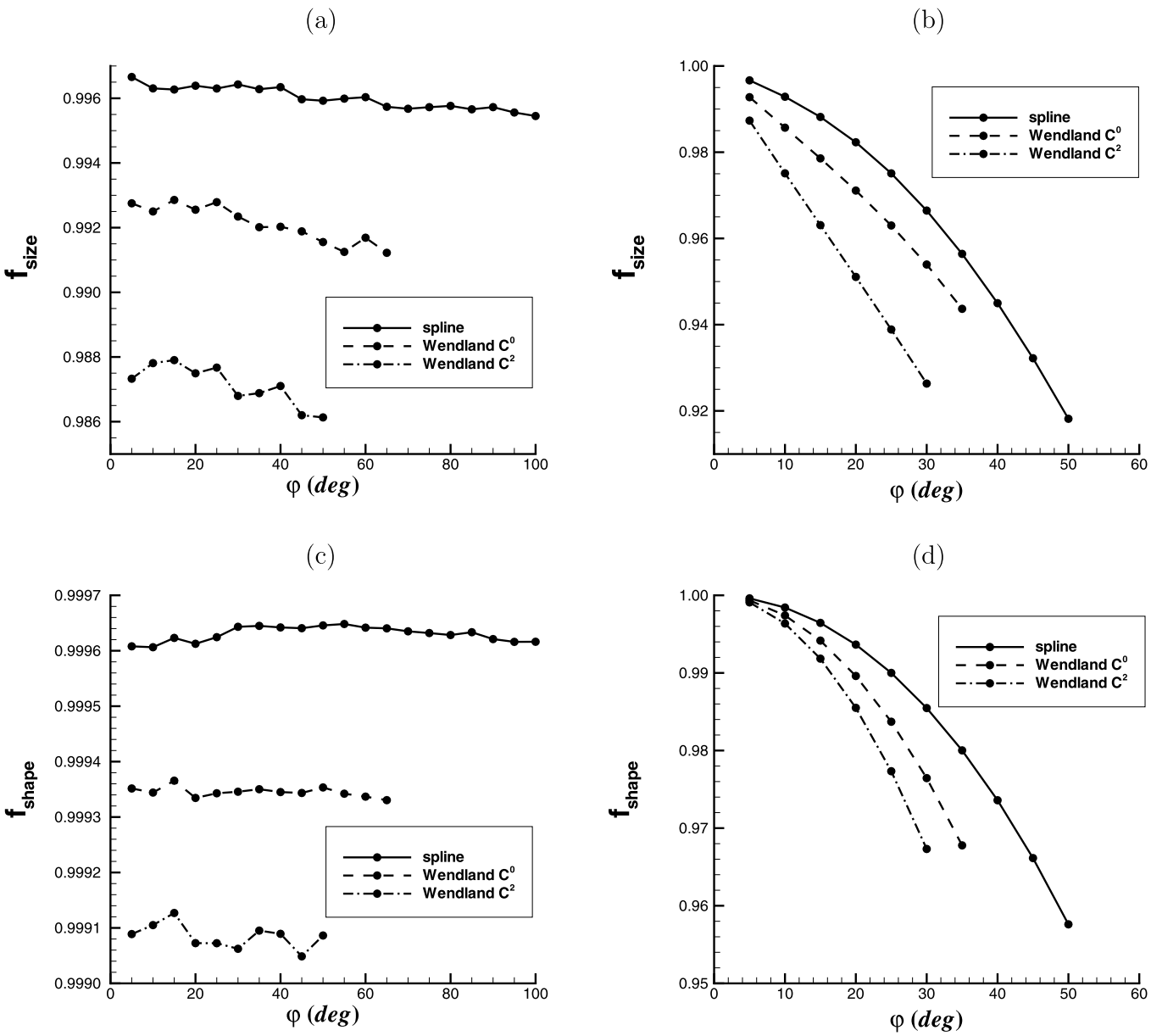

Fig. 13. NACA0012 validation test. Mean values of quality parameters for torsion test: (a) and (c) multi-step deformation; (b) and (d) one step deformation.

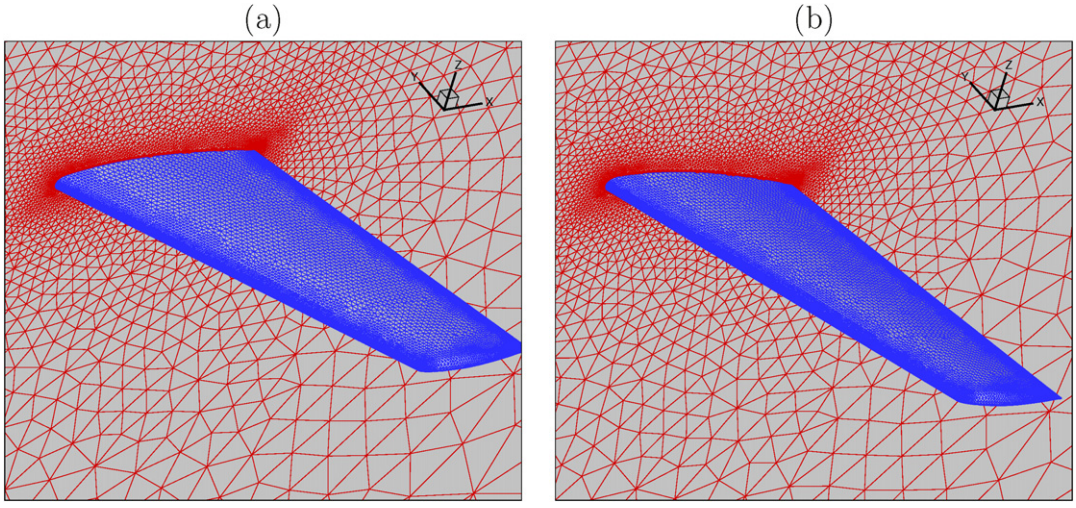

Fig. 14. ONERA M6: (a) original mesh; (b) deformed mesh after a wing rotation of 10°.

Some examples of the final deformed mesh are shown in Figs. 9–11. In particular, Figs. 11(a) and 11(b) show a detail of the deformed mesh in a section close to the wing tip located at 80% of the span and the wing leading edge respectively.

The maximum value of the achieved deformation, shown in Table 2, is limited by the conservation of the quality of the deformed mesh, according to the parameters described in Section 4.2.1, or a maximum deformation of 100° (torsion) and 100%L (bending) when running the algorithm iteratively. This table also shows the results obtained with two different RBF functions: Wendland $\mathcal{C}^0$ and Wendland $\mathcal{C}^2$. These results illustrate the advantage of performing the deformation in small steps, which allows to reach higher final values of deformation. The quality parameters are depicted in Figs. 12 and 13 which show that in the worst case, the mesh quality parameter is about 0.92, independently of the chosen interpolant function. This highlights the ability of the *MeshMove* tool to handle large mesh deformations in a smooth and robust manner.

Another important feature of this method is its low computational cost in both, memory and time. The preprocessing, which includes octree generation as well as the definition of the interpolation domains along with their corresponding deformation order, uses only 3% of the total execution time (see Table 3).

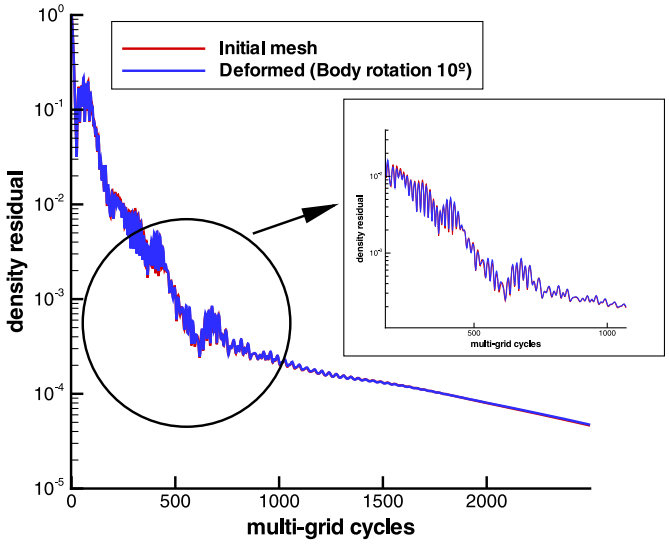

Fig. 15. ONERA M6. Comparison between the convergence evolution of the initial and deformed meshes.

4.2.3. Viscous mesh

The ONERA M6 wing [22] configuration has been selected to demonstrate the ability of the developed tool to carry out deformations of Navier–Stokes meshes without significant quality degradation. An unstructured hybrid mesh with 1.5×10^6 points (2.5×10^6 tetrahedrons and 2.1×10^6 prisms) has been generated for turbulent computations at subsonic flow conditions (Mach $= 0.2$, Re $= 11.2 \times 10^6$ and $\alpha = 10°$). In order to assess the possible quality degradation induced by the mesh deformation tool, two separate computations of the same configuration will be carried out, one over the original mesh and another over a deformed mesh. Hence, the 1st computation uses the original mesh with the angle of attack $\alpha = 10°$. In a second computation, the wing is rotated as a rigid body by 10° about the y-axis, the rest of the mesh is deformed accordingly, and the angle of attack is prescribed to $\alpha = 0°$ (see Fig. 14).

Both computations have been carried out using the DLR-TAU code [24] with the Spalart–Allmaras turbulence model. Fig. 15 shows the convergence of the density residual for both computations showing that the convergence has not been deteriorated in the deformation process, owing to the quality of the mesh is preserved throughout the deformation process (i.e., mean values: $f_{size} = 0.984$ and $f_{shape} = 0.995$). The same conclusion is obtained from the comparison of the global force coefficients (Fig. 16). In order to compare the solutions, the pressure coefficient distributions are drawn for three arbitrary sections (located at the wing inboard, mid and outboard) for both solutions. Fig. 17 shows the pressure distributions for these three sections. The differences in the pressure values between the initial and deformed grid are almost negligible. In conclusion, the comparison between both solutions

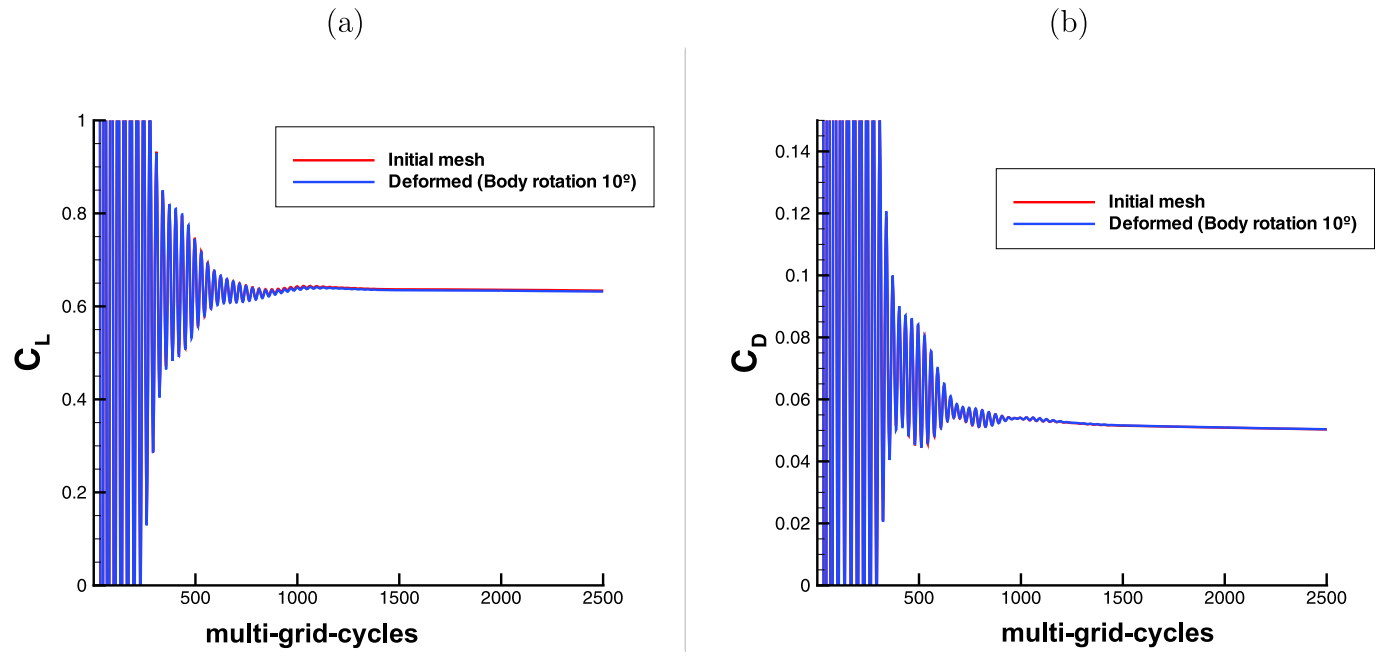

Fig. 16. ONERA M6. Initial and deformed mesh convergence histories. (a) Lift coefficient convergence evolution; (b) drag coefficient evolution.

Table 3
Breakdown of CPU time using local interpolation strategy.

Mesh	N_s	N_v	Preprocess time [s]	Evaluation time [s]	# External cubes	# Interpolation domains
cylinder	3131	340194	8.45	244.24	12622	10156
cylinder	4681	340194	8.71	244.47	12818	10352
cylinder	6231	340194	8.86	260.88	12832	10366
cylinder	9246	340194	11.45	271.35	12832	10366
cylinder	12261	340194	11.14	281.98	12832	10366
cylinder	18361	340194	11.32	303.73	12832	10366
cylinder	24461	340194	11.97	331.46	12881	10401
cylinder	30561	340194	12.10	361.31	13000	10483
NACA0012	34007	180101	4.90	77.46	10130	8582
ONERA M6	43218	1528078	330.64	893.72	67607	60831

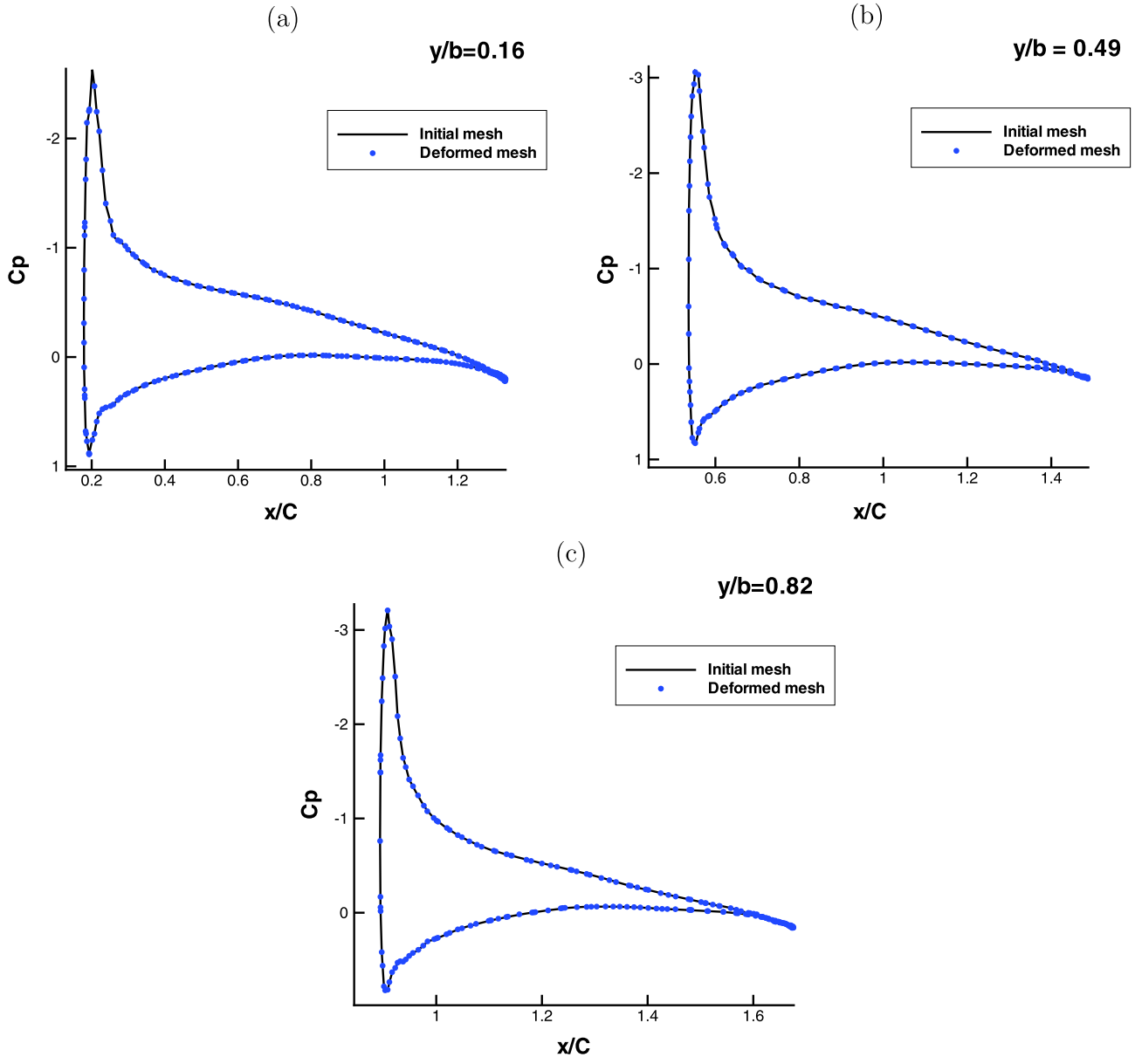

Fig. 17. ONERA M6. Pressure coefficient distributions at three sections of the wing.

shows an excellent agreement due to the ability of the present method to maintain the quality of the mesh in the deformation process.

Again, in Table 3 is shown the preprocessing and interpolation time for this test case. As it was already mentioned, in this case, the number of interpolation domains and external cubes, hence the computational time, is larger than the comparatively similar inviscid test case, due to the clustering of nodes close to the surface, which dramatically increase the number of domains.

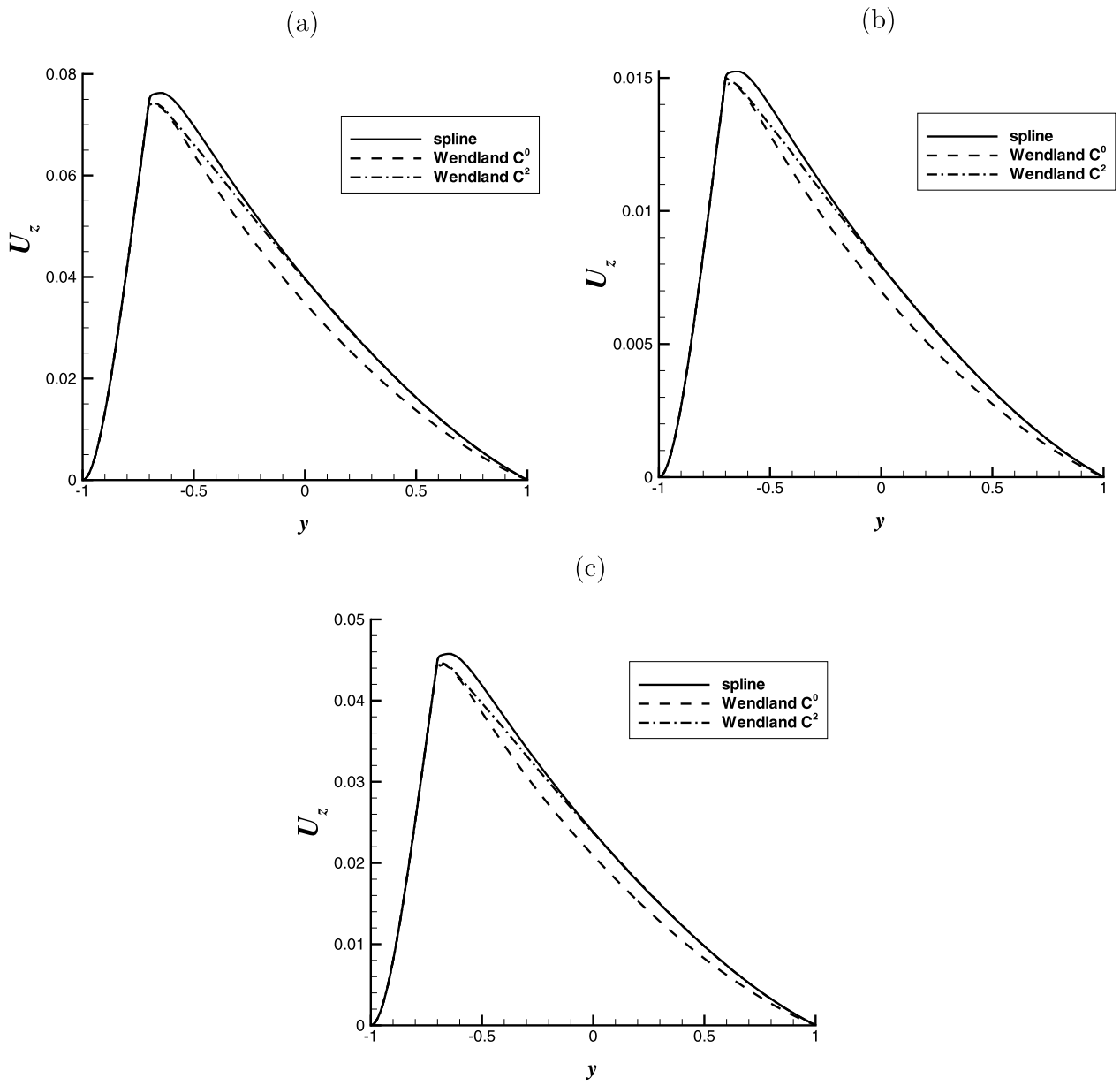

Fig. 18. NACA0012. Comparison of RBF interpolation near the wing leading edge: (a) Bending 5%L; (b) bending 15%L; (c) bending 25%L.

4.3. Comparison of interpolation functions

While this tool was originally conceived to be used with volume spline interpolation functions, its structure makes it very easy to incorporate other interpolation functions which can be used for further testing. *MeshMove* allows the user to specify the interpolator which best suits his requirements.

Along these lines, a set of tests has been carried out using, besides the volume splines interpolators, RBF functions with compact support defined in Table 1 to transfer torsion and bending deformations described in the previous section. The utility of such functions is well established (Wendland [27]). The aim of the present test being therefore simply to compare their performance, within the developed tool, with the volume spline interpolator.

The RBF functions have been incorporated by considering a variable support radius on each interpolation domain, the value of which is given by the maximum distance among the interpolation centers and the evaluation nodes that a domain contains. The results obtained show that the deformation transfer is essentially independent of the interpolation function, as was to be expected given the local character of the strategy implemented in the *MeshMove* tool. Fig. 18 shows a comparison for various interpolation functions of the vertical displacement of the wing's leading edge and its continuation into the fluid (volume) mesh, interpolated from the values of the displacement at the remaining surface nodes, for three vertical displacements at wing tip, η, representative of the tests carried out.

As can be seen from Fig. 18, each of the three tested functions give the same deformation near the wing's leading edge. Within the volume mesh, maximum differences are not bigger than 6% of the maximum deformation at the wing tip. As previously pointed out by Beckert et al. [4], the evolution of the deformation obtained with volume spline functions is essentially the same as the one obtained with the compactly supported radial function Wendland $\mathcal{C}^2$.

5. Conclusions

To face the increasing need, in all the branches of engineering, to manipulate meshes the computational size of which lies close

to the current computational capabilities of personal computers, it is essential to develop methodologies based on the decomposition of a global problem into small, local problems of affordable computational cost. In the particular case of deformation transfer of an aerodynamic mesh to a volume mesh, the decomposition can be made independent of the topology of the involved meshes, making the definition of the local computational interpolation domains easier, as well as the subsequent parallelization of the problem.

In this work, a general interpolation tool (*MeshMove*) based on radial basis functions together with an advancing front strategy for moving computational 3D meshes have been developed. *MeshMove* tool can be straightforwardly applied to any kind of three dimensional data, it does not requires any order or indexes between the data and can be easily used to transfer deformations, loads or any other variables between different meshes. This numerical tool can easily manage any kind of meshes – multiblock structured or unstructured – in a very easy way, is robust, efficient, and preserves the quality of the original mesh even for very large deformations.

An added bonus of the tool is that it offers the possibility to control, via the user-provided parameter $N_{\max}$ (which is the maximum number of nodes contained within each cube of the octree), the size of the interpolation domains. Thus, if the number of centers is small, it is possible to carry out a global interpolation (with a single domain). Additionally, the present method can be easily parallelized because of the inherent domain decomposition strategy. Then, the real computational capabilities of the *MeshMove* tool will be shown.

Acknowledgements

This work has been partially supported by EADS-CASA, Spain. The authors want to thank Juan José Guerra for his valuable contribution in our discussions. They are also grateful to INTA for supply of NACA0012 and ONERA M6 meshes data, especially to Carlos Lozano for his useful comments.

References

[1] R.E. Bartels, Mesh strategies for accurate computation of unsteady spoiler and aeroelastic problems, Journal of Aircraft 37 (3) (2000) 521–525.
[2] J.T. Batina, Unsteady Euler airfoil solutions using unstructured dynamic meshes, AIAA Paper 89-0115, January 1989.
[3] J.T. Batina, Unsteady Euler algorithm with unstructured dynamic mesh for complex-aircraft aerodynamic analysis, AIAA Journal 29 (3) (1992) 327–333.
[4] A. Beckert, H. Wendland, Multivariate interpolation for fluid–structure-interaction problems using radial basis functions, Aerospace Science and Technology 5 (3) (2001) 125–134.
[5] M. Buhmann, Radial basis functions: the state-of-the-art and new results, Acta Numerica 9 (2000) 1–37.
[6] M. Cordero-Gracia, P. Ripollés, E. Valero, M. Gomez, A volume spline interpolation tool for elastomechanical and aerodynamic data transfer problems, in: Proceedings of the IASTED International Conference on Applied Simulation and Modelling, 2004, pp. 265–269.
[7] A. de Boer, M.S. van der Schoot, H. Bijl, Mesh deformation based on radial basis function interpolation, Computers and Structures 85 (2007) 784–795.
[8] C. Farhat, C. Degand, B. Koobus, M. Lesoinne, Improved method of spring analogy for dynamic unstructured fluid meshes, AIAA Paper 98-2070, April 1998.
[9] X.-W. Gao, P.-C. Chen, L. Tang, Deforming mesh for computational aeroelasticity using a nonlinear elastic boundary element method, AIAA Journal 40 (8) (2002) 1512–1517.
[10] M.H.L. Hounjet, J.J. Meijer, Evaluation of elastomechanical and aerodynamic data transfer methods for non-planar configurations in computational aeroelastic analysis, NLR-TP-95690-U, June 1995.
[11] S. Jakobsson, O. Amoignon, Mesh deformation using radial basis functions for gradient based aerodynamic shape optimization, Technical Report FOI-R-1784-SE, FOI, 2005.
[12] P.M. Knupp, Algebraic mesh quality metrics, SIAM Journal on Scientific Computing 23 (2001) 193–218.
[13] P.M. Knupp, A method for hexaedra mesh shape optimization, International Journal for Numerical Methods in Engineering 58 (2003) 319–332.
[14] D.E. Knuth, The Art of Computer Programming, vol. 3, Addison-Wesley, USA, 1997.
[15] X. Liu, N. Qin, H. Xia, Fast dynamic mesh deformation based on Delaunay graph mapping, Journal of Computational Physics 211 (2006) 405–423.
[16] R. Löhner, Finite elements in CFD: mesh generation, adaptivity and parallelization, AGARD-R-787, No. 8, 1992.
[17] D.G. Martineau, J.M. Georgala, A mesh movement algorithm for high quality generalised meshes, AIAA Paper 2004-0614.
[18] A.K. Michler, Aircraft control surface deflection using RBF-based mesh deformation, in: V European Conference on Computational Fluid Dynamics, 2010, pp. 1–20.
[19] S.A. Morton, R.B. Melville, M.R. Visbal, Accuracy and coupling issues of aeroelastic Navier–Stokes solutions on deforming meshes, Journal of Aircraft 35 (5) (1998) 798–805.
[20] T.C.S. Rendall, C.B. Allen, Unified fluid–structure interpolation and mesh motion using radial basis functions, International Journal for Numerical Methods in Engineering 74 (2008) 1519–1559.
[21] T.C.S. Rendall, C.B. Allen, Parallel efficient mesh motion using radial basis functions with application to multi-bladed rotors, International Journal for Numerical Methods in Engineering 81 (2010) 89–105, 2010.
[22] V.J. Schmitt, F. Charpin, Pressure distributions on the ONERA-M6-wing at transonic Mach numbers, Experimental Data Base for Computer Program Assessment, AGARD-R-138, May 1979.
[23] M.J. Smith, D.H. Hodges, C.E. Cesnik, An evaluation of computational algorithms to interface between CFD and CSM methodologies, NASA Technical Reports, AD-A313520, 1995.
[24] TAU-code user guide, Release 2009.1.0, March 30, 2009.
[25] H.M. Tsai, S.F. Wong, J. Cai, Y. Zhu, F. Liu, Unsteady flow calculations with a parallel multiblock moving mesh algorithm, AIAA Journal 39 (6) (2001) 1021–1029.
[26] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, Advances in Computational Mathematics 4 (1995) 389–396.
[27] H. Wendland, Hybrid methods for fluid–structure-interaction problems in aeroelasticity, in: Meshfree Methods for Partial Differential Equations IV, Springer, Berlin/Heidelberg, 2008, pp. 335–358.
[28] http://www.netlib.org/lapack/.